(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112602058 B
(45)授权公告日 2024.03.15

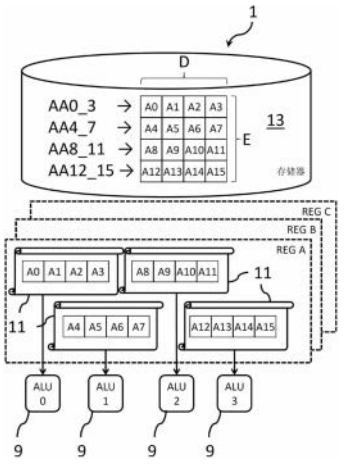
(21)申请号 201980053161.8
(22)申请日 2019.05.21
(65)同一申请的已公布的文献号
申请公布号 CN 112602058 A
(43)申请公布日 2021.04.02
(30)优先权数据
1855998 2018.06.29 FR
(85)PCT国际申请进入国家阶段日
2021.02.07
(86)PCT国际申请的申请数据
PCT/FR2019/051155 2019.05.21
(87)PCT国际申请的公布数据
WO2020/002782 FR 2020.01.02
(73)专利权人 弗索拉公司
地址 法国默东拉福雷特
(72)发明人 克哈莱德·玛来吉
特朗格-邓格·恩古延
朱利恩·斯奇米特
皮埃尔-伊曼纽尔·伯纳德
(74)专利代理机构 北京国昊天诚知识产权代理有限公司 11315
专利代理师 南霆 李有财
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)
G06F 15/80(2006.01)
(56)对比文件
CN 104395876 A,2015.03.04
CN 1489732 A,2004.04.14
CN 1713133 A,2005.12.28
US 5513366 A,1996.04.30
WO 1997032249 A1,1997.09.04
US 2003014457 A1,2003.01.16
EP 2144158 A1,2010.01.13
US 2010312997 A1,2010.12.09
US 2006047937 A1,2006.03.02
US 2011026846 A1,2011.02.03
GB 0409815 D0,2004.06.09
严伟,龚幼民.高性能数字信号处理器的设计.微处理机.2004,10-15.
审查员 彭玉芝
权利要求书2页 说明书12页 附图4页

(54)发明名称

处理器存储器存取

(57)摘要

本发明涉及一种计算装置,其包括:多个ALU(9);一组寄存器(11);存储器(13);存储器接口,其在所述寄存器(11)和所述存储器(13)之间;控制单元(5),其通过生成以下各项来控制所述ALU(9):至少一个循环i,其包含借助于算术逻辑单元(9)实施至少一个第一计算操作和从所述存储器(13)将第一数据集(AA4_7;BB4_7)下载到至少一个寄存器(11);以及至少一个循环ii,其在所述至少一个循环i之后,包含借助于算术逻辑单元(9)实施第二计算操作,针对所述第二计算操作,所述第一数据集(AA4_7;BB4_7)的至少一部分(A4;B4)形成至少一个操作数。



CN 112602058 B

1.一种计算装置(1),其包括:

-多个算术逻辑单元(9);

-一组寄存器(11),其能够将操作数类型的数据供应到所述算术逻辑单元(9)的输入,且能够被供应来自所述算术逻辑单元(9)的输出的数据;

-存储器(13);

-存储器接口(15),借助于所述存储器接口(15)在所述寄存器(11)和所述存储器(13)之间传输和路由数据(A0、A15);

-控制单元(5),其被配置成根据处理链微架构控制所述算术逻辑单元(9),使得所述算术逻辑单元(9)彼此并行地执行计算操作,所述控制单元(5)进一步被设计成借助于所述存储器接口(15)控制存储器存取操作,

由所述控制单元(5)控制的控制操作生成:

-至少一个循环i,其包含借助于算术逻辑单元(9)并行实施至少一个第一计算操作和从所述存储器(13)将第一数据集(AA4_7;BB4_7)下载到至少一个寄存器(11),所述第一数据集(AA4_7;BB4_7)的至少一部分(A4;B4)在循环i期间不被算术逻辑单元(9)的任何计算操作使用;

-至少一个循环ii,其在所述至少一个循环i之后,包含借助于算术逻辑单元(9)实施第二计算操作,针对所述第二计算操作,所述第一数据集(AA4_7;BB4_7)的所述至少一部分(A4;B4)形成至少一个操作数。

2.根据权利要求1所述的装置,其特征在于,所述控制单元(5)进一步被配置成,在控制所述算术逻辑单元和所述存储器存取操作之前,实施识别算法,以用于基于所述至少一个循环ii期间待实施的所述第二计算操作识别所述至少一个循环i期间待下载的所述第一数据集(AA4_7;BB4_7)。

3.根据权利要求1所述的装置,其特征在于,所述控制单元(5)被配置成实施彼此分离的两个循环i,使得彼此分离的两个第一数据集(AA4_7、BB4_7)下载到至少一个寄存器(11),所述两个第一数据集(AA4_7、BB4_7)中的每一个的至少一部分(A4、B4)形成用于所述至少一个循环ii的所述第二计算操作的操作数。

4.根据权利要求1所述的装置,其特征在于,所述控制单元(5)被配置成实施彼此分离的多个循环ii,且使得形成用于循环ii的所述第二计算操作的至少一个操作数的所述第一数据集(AA4_7)的部分(A4;A5;A6;A7)在所述多个循环ii之间不同。

5.根据权利要求1所述的装置,其特征在于,所述控制单元(5)被配置成执行一系列至少一个循环i和一个循环ii的至少两个迭代,所述两个迭代至少部分叠加使得第一次迭代的至少一个循环ii形成随后迭代的循环i。

6.根据权利要求1所述的装置,其特征在于,所述控制单元(5)被配置成在第一循环i之前设置初始化阶段,所述初始化阶段包含从所述存储器(13)向至少一个寄存器(11)下载至少一个数据集(AA0_3;BB0_3),所述至少一个数据集(AA0_3;BB0_3)形成用于所述第一循环i的所述第一计算操作的操作数。

7.根据权利要求1所述的装置,其特征在于,所述控制单元(5)进一步被设计成借助于所述存储器接口(15)控制所述存储器存取操作,使得所述控制操作生成:

-在循环i期间,由多个算术逻辑单元(9)实施多个第一计算操作;

-在循环ii期间,由多个算术逻辑单元(9)实施多个第二计算操作,

选择每个待下载数据集的数据的分组以便匹配计算操作到多个算术逻辑单元(9)中的每一个的指派的分配,使得所述算术逻辑单元(9)具有同步、异步或混合操作。

8.一种由计算装置(1)的控制单元(5)实施的数据处理方法,所述装置(1)包括:

-多个算术逻辑单元(9);

-一组寄存器(11),其能够将操作数类型的数据供应到所述算术逻辑单元(9)的输入,且能够被供应来自所述算术逻辑单元(9)的输出的数据;

-存储器(13);

-存储器接口(15),借助于所述存储器接口(15)在所述寄存器(11)和所述存储器(13)之间传输和路由数据(A0、A15);

-所述控制单元(5),其被配置成根据处理链微架构控制所述算术逻辑单元(9),使得所述算术逻辑单元(9)彼此并行地执行计算操作,所述控制单元(5)进一步被设计成借助于所述存储器接口(15)控制存储器存取操作;

所述方法至少包括:

-生成循环i,所述循环i包含借助于算术逻辑单元(9)并行实施至少一个第一计算操作和从所述存储器(13)将第一数据集(AA4_7;BB4_7)下载到至少一个寄存器(11),所述第一数据集(AA4_7;BB4_7)的至少一部分(A4;B4)在循环i期间不被算术逻辑单元(9)的任何计算操作使用;

-生成循环ii,所述循环ii在所述循环i之后,包含借助于算术逻辑单元(9)实施第二计算操作,针对所述第二计算操作,所述第一数据集(AA4_7;BB4_7)的所述至少一部分(A4;B4)形成至少一个操作数。

9.一种非暂时性计算机可读记录介质,在其上面记录程序,所述程序用于当此程序由处理器执行时实施根据权利要求8所述的方法。

处理器存储器存取

技术领域

[0001] 本发明涉及处理器的领域,且涉及其与存储器单元的交互。

背景技术

[0002] 常规地,计算装置包括一个或多个处理器的集合。每一处理器包括一个或多个处理单元或PU。每一PU包括称为算术逻辑单元或ALU的一个或多个计算单元。为了具有高性能计算装置,也就是说,快速的以便执行计算操作的计算装置,常规做法是提供高数目的ALU。因此,ALU能够并行地(即,同时)处理操作。因而,时间单位为计算循环。因此,通常依据每计算循环能够执行的操作的数目来量化计算装置的计算能力。

[0003] 然而,如果装置的与ALU交互的元件未被设计(设定尺寸)成与期望同时操作的ALU的数目一致,则具有高数目的ALU并不适当,乃至是不必要的。换句话说,如果存在高数目的ALU,则ALU的环境的配置可以是限制装置的功率的准则。确切地说,所述装置包括存储器组合件,其本身包括一个或多个存储器单元,每一存储器单元具有能够永久地存储计算数据的固定数目的存储器位置。在计算处理操作期间,ALU在输入处从存储器单元接收数据,且在输出处供应数据,所述数据则被存储在存储器单元上。因而应理解,除ALU的数目外,存储器单元的数目是确定装置的计算能力的另一准则。

[0004] 数据在两个方向上通过装置的总线在ALU和存储器单元之间路由。本文中使用的术语“总线”一般是指用于传递数据的系统(或接口,其包含硬件(接口电路)和管控交换的协议。总线自身传输的数据、地址和控制信号。每一总线本身还具有硬件和软件限制,使得数据的路由受到限制。确切地说,总线具有存储器单元侧上的有限数目的端口和ALU侧上的有限数目的端口。在本文中,认为存储器单元为单端口,也就是说,与被称为“双端口”存储器(就表面而言较昂贵,且需要较大双控制总线用于读取和写入)的存储器相比,在不同循环期间实施读取和写入操作。因此,在计算循环期间,可经由总线在单个方向中(在“读取”模式中或在“写入”模式中)存取存储器位置。此外,在计算循环期间,存储器位置仅可由单个ALU存取。作为变型,可利用被称为“双端口”存储器的存储器来实施所提议的技术方案。在此些实施例中,可在同一个计算循环期间实施读取和写入操作。

[0005] 在总线和ALU之间,计算装置通常包括一组寄存器和本地存储器单元,其可视为与上述存储器单元分离的存储器。为了便于理解,此处绘示既定存储这样的数据的“寄存器”和既定存储存储器地址的“本地存储器单元”之间的区别。向每一寄存器指派PU的ALU。向PU指派多个寄存器。寄存器的存储容量相比于存储器单元非常有限,但其内容可由ALU直接存取。

[0006] 为了执行计算操作,每一ALU通常必须首先获得计算操作的输入数据,通常是基础计算操作的两个操作数。因此实施经由总线的相应存储器位置上的“读取”操作以便将所述两个操作数中的每一个导入到寄存器上。ALU接着自身基于来自寄存器的数据且通过将结果以数据项的形式导出到寄存器上来执行计算操作。最后。实施“写入”操作以便将计算操作的结果记录在存储器位置中。在此写入操作期间,存储在寄存器上的结果经由总线记录

在存储器位置中。据推断,操作中的每一个消耗一个或多个计算循环。

[0007] 在已知计算装置中,通常尝试在同一个计算循环期间执行多个操作(或多个指令),以便减小计算循环的总数目且因此提高效率。接着参考并行“处理链”或“管线”。然而,操作之间常常存在许多相互相依性。举例来说,只要操作数尚未读取且它们不可在用于ALU的寄存器上存取,就不可能执行基础计算操作。因此,实施处理链涉及检查操作(指令)之间的相互相依性,这很复杂且因此代价很高。

[0008] 通常在同一个计算循环期间实施多个独立的操作。通常,对于给定ALU且在同一个计算循环期间,有可能执行计算操作和读取或写入操作。相比之下,对于给定ALU且在同一个计算循环期间,不可能同时执行读取操作和写入操作(在单端口存储器单元的情况下)。另一方面,存储器存取操作(总线)并不使得有可能在同一个计算循环期间且针对给定存储器位置为彼此分离的两个ALU执行读取或写入操作。

[0009] 为了确保每一ALU尽可能活跃(无损失的计算循环),因此本能地尝试实现在每一计算循环处可针对ALU中的每一个存取三个存储器位置的情形:两个存储器位置既定将操作数供应到ALU的两个输入(读取模式),且一个存储器位置用于接收来自ALU的基础计算操作结果(写入模式)。因此,两个读取操作被选择以便获得(存储在寄存器上)随后计算循环期间待实施的基础计算操作所需的操作数。为了改进计算能力,因此本能地提供高数目的ALU和成比例的数目的存储器位置(例如,至少比ALU多三倍的存储器位置)。

[0010] 然而,ALU的数目和存储器单元的数目的增加会增加这两种类型的元件之间的交互的复杂性。增加装置的ALU和能够连接到其上的存储器单元的数目导致总线的复杂性非线性地增加。增加ALU的数目和存储器单元的数目因此既复杂又昂贵。

[0011] 本发明旨在改进这种情形。

发明内容

[0012] 提出一种计算装置,其包括:

[0013] -多个算术逻辑单元;

[0014] -一组寄存器,其能够将操作数类型的数据供应到所述算术逻辑单元的输入,且能够被供应来自所述算术逻辑单元的输出的数据;

[0015] -存储器;

[0016] -存储器接口,借助于其在寄存器和存储器之间传输和路由数据;

[0017] -控制单元,其被配置成根据处理链微架构控制算术逻辑单元,使得算术逻辑单元彼此并行地执行计算操作,所述控制单元进一步被设计成借助于所述存储器接口控制存储器存取操作。控制操作生成:

[0018] -至少一个循环i,其包含借助于算术逻辑单元实施至少一个第一计算操作和将第一数据集从存储器下载到至少一个寄存器;

[0019] -至少一个循环ii,其在所述至少一个循环i之后,包含借助于算术逻辑单元实施第二计算操作,针对所述第二计算操作,第一数据集的至少一部分形成至少一个操作数。

[0020] 此装置使得有可能在单个操作中从存储器单元读取数据集,且将所述数据暂时存储在寄存器上。计算循环t中读取的所有数据不可全部在紧接的后续计算循环t+1中使用。在至少一些情况下,所读取数据当中的一些数据在所述后续计算循环t+1期间是不必要的,

但在另一随后计算循环t+1+n期间使用,而不必执行额外读取操作,且因此不会消耗额外计算循环。

[0021] 在使用计算装置的并行数据处理的领域中,常见方法是调度专用存储器存取操作来读取待实施的下一基础计算操作中所需的数据中的每一个,且仅下一基础计算操作所需的那些数据。此常见方法可称为“准时制存储器存取(just-in-time memory access)”。在此常见方法中,认为读取(且存储在寄存器上)并非立即需要的数据项是不必要的。因此,每一存储器存取操作在时间上位于基础计算操作本身(必不可少的)之前,但直接且仅基于下一基础计算操作而调度。申请人已经违背领域中的先验知识而实施一种不同的方法。

[0022] 因此,申请人提出一种方法,其中,在每一读取操作后,所读取数据的数目大于实施下一计算操作精确地必需的数据的数目。相对来说,此方法可称为“临时存储器存取”。因而,有可能来自所读取数据当中的一个数据项用于将来计算操作,紧接在读取操作之后实施的计算操作除外。在此些情况下,已经在单个存储器存取操作期间获得必需的数据(其中存储器的带宽增加),而常见方法原本一直需要至少两个单独的存储器存取操作。因此,至少在某些情况下,申请人提出的方法的效果为:减少用于存储器存取操作的计算循环的消耗,且因此使得有可能改进装置的效率。长时间(多个连续计算循环)内,存储器存取操作的数目(读取模式中和/或写入模式中)减小。

[0023] 此方法不排除损失:即使在已在计算操作中使用之前,被读取且存储在寄存器上的一些数据也可能丢失(被接着存储在同一寄存器上的其它数据擦除)。然而,经过大量计算操作和计算循环,申请人观察到性能的改进,包含在不选择所读取数据集的情况下。换句话说,即使在不选择所读取数据(或随机选择)的情况下,此方法也使得有可能与常见方法相比统计上改进计算装置的效率。

[0024] 根据另一方面,提出一种由计算装置的控制单元实施的数据处理方法,所述装置包括:

[0025] -多个算术逻辑单元;

[0026] -一组寄存器,其能够将操作数类型的数据供应到所述算术逻辑单元的输入,且能够被供应来自所述算术逻辑单元的输出的数据;

[0027] -存储器;

[0028] -存储器接口,借助于其在寄存器和存储器之间传输和路由数据;

[0029] -控制单元,其被配置成根据处理链微架构控制算术逻辑单元,使得算术逻辑单元彼此并行地执行计算操作,所述控制单元进一步被设计成借助于所述存储器接口控制存储器存取操作;

[0030] 所述方法至少包括:

[0031] -生成循环i,所述循环i包含借助于算术逻辑单元实施至少一个第一计算操作和将第一数据集从存储器下载到至少一个寄存器;

[0032] -生成循环ii,所述循环ii在循环i之后,包含借助于算术逻辑单元实施第二计算操作,针对所述第二计算操作,第一数据集的至少一部分形成至少一个操作数。

[0033] 根据另一方面,提出一种计算机程序,确切地说编译计算机程序,其包括用于当此程序由处理器执行时实施如本文所定义的方法的全部或部分的指令。根据另一方面,提出一种在上面记录此程序的非暂时性计算机可读记录介质。

[0034] 可以任选地实施以下特征。可以彼此独立地或彼此组合地实施所述特征：

[0035] -控制单元进一步被配置成，在控制算术单元和存储器存取操作之前，实施识别算法，以用于基于所述至少一个循环ii期间待实施的第二计算操作识别所述至少一个循环i期间待下载的第一数据集。这使得有可能基于待执行的计算操作调适所读取数据，且因此改进所下载的第一数据集中的数据的相关性。

[0036] -控制单元被配置成实施彼此分离的两个循环i，使得彼此分离的两个第一数据集下载到至少一个寄存器，两个第一数据集中的每一个的至少一部分形成用于所述至少一个循环ii的第二计算操作的操作数。这使得有可能在所述至少两个循环期间，组合计算操作和存储器存取操作。因此，在两个循环i结束时，可能已下载随后计算操作所需的所有操作数。

[0037] -控制单元被配置成实施彼此分离的多个循环ii，且使得形成用于循环ii的第二计算操作的至少一个操作数的第一数据集的所述部分在所述多个循环ii之间不同。这使得有可能针对单个所下载数据集实施多个基础计算操作。

[0038] -控制单元被配置成执行一系列至少一个循环i和一个循环ii的至少两个迭代，所述两个迭代至少部分叠加使得第一次迭代的至少一个循环ii形成随后迭代的循环i。这使得有可能在尤其有限数目的计算循环内执行计算操作和存储器存取操作，且因此进一步提高效率。

[0039] -控制单元被配置成在第一循环i之前设置初始化阶段，所述初始化阶段包含从存储器向至少一个寄存器下载形成用于所述第一循环i的第一计算操作的操作数的至少一个数据集。这使得有可能使方法初始化，以便接着根据需要多次针对待处理的数据集中的每一个重复循环i和ii。

[0040] -控制单元进一步被设计成借助于所述存储器接口控制存储器存取操作，使得所述控制操作生成：

[0041] -在循环i期间，多个第一计算操作由多个算术逻辑单元的实施；

[0042] -在循环ii期间，多个第二计算操作由多个算术逻辑单元的实施，

[0043] 选择每待下载数据集的数据的分组以便匹配计算操作到所述多个算术逻辑单元中的每一个的指派的分配，使得所述算术逻辑单元具有同步、异步或混合操作。这使得有可能通过基于待执行的处理操作和可用资源调适ALU的协调操作来进一步改进效率。

附图说明

[0044] 在阅读以下详细描述和分析附图后，本发明的其它特征、细节和优点将变得显而易见，附图中：

[0045] -图1展示根据本发明的计算装置的架构；

[0046] -图2是根据本发明的计算装置的架构的部分描绘；以及

[0047] -图3展示根据本发明的存储器存取操作的一个实例；以及

[0048] -图4是来自图3的实例的变型。

具体实施方式

[0049] 图式和下文的描述基本上含有特定性质的元件。因此，其将不仅能够有助于对本

发明的更好理解,而且有助于在适当时对本发明进行定义。

[0050] 图1展示计算装置1的一个实例。装置1包括一个或多个处理器3的集合,有时称为中央处理单元或CPU。所述处理器3的集合包括至少一个控制单元5和至少一个处理单元7或PU 7。每一PU 7包括一个或多个计算单元,称为算术逻辑单元9或ALU 9。在本文描述的实例中,每一PU 7进一步包括一组寄存器11。装置1包括能够与所述处理器3集合交互的至少一个存储器13。为此,装置1进一步包括存储器接口15或"总线"。

[0051] 在本文中,认为存储器单元为单端口,也就是说,与被称为"双端口"存储器(就表面而言较昂贵,且需要较大双控制总线用于读取和写入)的存储器相比,在不同循环期间实施读取和写入操作。作为变型,可利用被称为"双端口"存储器的存储器来实施所提议的技术方案。在此些实施例中,可在同一个计算循环期间实施读取和写入操作。

[0052] 图1展示三个PU 7:PU 1、PU X和PU N。仅详细展示PU X的结构以便简化图1。然而,PU的结构彼此类似。在一些变型中,PU的数目不同。装置1可包括单个PU、两个PU或超过三个PU。

[0053] 在本文描述的实例中,PU X包括四个ALU:ALU X.0、ALU X.1、ALU X.2和ALU X.3。在一些变型中,PU可包括彼此不同的和/或除四个以外的若干ALU,包含单个ALU。每一PU包括一组寄存器11,此处为指派到每一ALU的至少一个寄存器11。在本文描述的实例中,PU X包括每ALU单个寄存器11,也就是说,四个寄存器,参考为REG X.0、REG X.1、REG X.2和REG X.3,且分别指派到ALU X.0、ALU X.1、ALU X.2和ALU X.3。在一些变型中,每一ALU被指派多个寄存器11。

[0054] 每一寄存器11能够将操作数数据供应到所述ALU 9的输入,且能够被供应来自所述ALU 9的输出的数据。此外,每一寄存器11能够存储来自存储器13的借助于总线15经由称为"读取"操作的操作获得的数据。此外,每一寄存器11能够向存储器13且借助于总线15经由称为"写入"操作的操作传输所存储数据。通过从控制单元5控制存储器存取操作来管理读取和写入操作。

[0055] 控制单元5强加每一ALU 9执行基础计算操作的方式,确切地说,其次序,且向每一ALU 9指派待执行的操作。在本文描述的实例中,控制单元5被配置成根据处理链微架构控制ALU 9,使得ALU 9彼此并行地执行计算操作。举例来说,装置1具有单指令流和多数据流架构,称为SIMD,代表"单指令多数据";和/或多指令流和多数据流架构,称为MIMD,代表"多指令多数据"。另一方面,控制单元5进一步被设计成借助于存储器接口15控制存储器存取操作,且确切地说,在此情况下为读取和写入操作。两种类型的控制(计算和存储器存取)由图1中的虚线箭头展示。

[0056] 现在参考图2,其中展示单个ALU Y。数据传输由实线箭头展示。因为数据是逐步传输的,所以应了解,图2不一定展示具有同时数据传输的时间t。相比之下,为了从寄存器11向ALU 9传输数据项,举例来说,必须预先(在此情况下)经由存储器接口15(或总线)将所述数据项从存储器13传输到所述寄存器11。

[0057] 在图2的实例中,分别参考为REG Y.0、REG Y.1和REG Y.2的三个寄存器11被指派参考为ALU Y的ALU。每一ALU 9具有至少三个端口,确切地说,两个输入和一个输出。对于每一操作,分别由第一和第二输入接收至少两个操作数。经由输出传输计算操作的结果。在图2所示的实例中,输入处接收的操作数分别从寄存器REG Y.0以及从寄存器REG Y.2发起。将

计算操作的结果写入到寄存器REG Y.1。一旦其已写入到寄存器REG Y.1,则经由存储器接口15将结果(呈数据项的形式)写入到存储器13。在一些变型中,至少一个ALU可具有两个以上输入,且接收两个以上操作数用于计算操作。

[0058] 每一ALU 9可执行:

[0059] -对数据的整数算术运算(加法、减法、乘法、除法等);

[0060] -对数据的浮点算术运算(加法、减法、乘法、除法、求逆、平方根、对数、三角函数等);

[0061] -逻辑运算(二的补码、“与(AND)”、“或(OR)”“异或”等)。

[0062] ALU 9并不直接彼此交换数据。举例来说,如果由第一ALU执行的第一计算操作的结果构成用于待由第二ALU执行的第二计算操作的操作数,则第一计算操作的结果应在能够由ALU 9使用之前至少写入到寄存器11。

[0063] 在一些实施例中,写入到寄存器11的数据进一步自动写入到存储器13(经由存储器接口15),即使获得所述数据项整体上仅为了充当操作数而非充当处理过程的结果。

[0064] 在一些实施例中,被获得以充当操作数且具有短暂相关性的数据(在处理操作结束时整体上没有意义的中间结果)不自动写入到存储器13,且可仅暂时存储在寄存器11上。举例来说,如果由第一ALU执行的第一计算操作的结果构成用于待由第二ALU执行的第二计算操作的操作数,则第一计算操作的结果应写入到寄存器11。接下来,所述数据项作为操作数直接从寄存器11传输到第二ALU。因而应理解,寄存器11到ALU 9的指派可随时间演变,且确切地说,从一个计算循环到另一计算循环演变。此指派可确切地说呈寻址数据的形式,这使得有可能始终定位数据项的位置,在寄存器11上或在存储器15中的位置处。

[0065] 在下文中,针对应用于计算数据的处理操作描述装置1的操作,所述处理操作分解为操作的集合,包含由多个ALU 9在由计算循环序列组成的时间周期期间并行地执行的计算操作。因而认为,ALU 9正根据处理链微架构操作。然而,由装置1实施且此处涉及的处理操作可本身构成较全局计算过程的部分(或子集)。此较全局过程可包括(在其它部分或子集中)由多个ALU例如在串行操作模式中或以级联方式非并行地执行的计算操作。

[0066] 操作架构(并行或串行)可以是恒定或动态的,例如由控制单元5强加(控制)。举例来说,架构变化可取决于待处理的数据,且取决于装置1的输入处接收的当前指令。架构的此动态调适可在编译阶段较早实施,方式是,在能够从源代码推断待处理的数据的类型和所述指令时基于待处理的数据的类型和所述指令调适由编译器生成的机器指令。此调适还可仅在装置1或执行常规机器代码的处理器处实施,且其被编程为取决于待处理的数据和当前接收的指令实施配置指令集。

[0067] 存储器接口15或“总线”在两个方向上在ALU 9和存储器15之间传输和路由数据。存储器接口15由控制单元5控制。控制单元5因此借助于存储器接口15控制对装置1的存储器13的存取。

[0068] 控制单元5以协调方式控制由ALU 9实施的(计算)操作和存储器存取操作。控制单元5所实施的控制包括实施分解为计算循环的操作序列。所述控制包括生成第一循环i和第二循环ii。第一循环i在时间上在第二循环ii之前。如以下实例中将更详细地描述,第二循环ii可紧接在第一循环i之后,或者第一循环i和第二循环ii可在时间上彼此间隔开,例如具有中间循环。

[0069] 第一循环i包括：

[0070] -借助于至少一个ALU 9实施第一计算操作;以及

[0071] -将第一数据集从存储器13下载到至少一个寄存器11。

[0072] 第二循环ii包括借助于至少一个ALU 9实施第二计算操作。第二计算操作可由与第一计算操作相同的ALU 9或由单独的ALU 9实施。第一循环i期间下载的第一数据集的至少一部分形成用于第二计算操作的操作数。

[0073] 现在参考图3。一些数据或数据块分别参考为A0到A15且存储于存储器13中。在所述实例中,认为数据A0到A15以四个的形式分组在一起,如下：

[0074] -参考为AA0_3的数据集,由数据A0、A1、A2和A3组成;

[0075] -参考为AA4_7的数据集,由数据A4、A5、A6和A7组成;

[0076] -参考为AA8_11的数据集,由数据A8、A9、A10和A11组成;以及

[0077] -参考为AA12_15的数据集,由数据A12、A13、A14和A15组成。

[0078] 作为变型,数据可以不同方式分组在一起,确切地说,以两个、三个或超过四个的群组(或“块”或“时隙”)的形式分组在一起。数据集可视为在单个读取操作期间借助于存储器接口15的单个端口可在存储器13上存取的一组数据。同样,数据集的数据可在单个写入操作期间借助于存储器接口15的单个端口写入到存储器13。

[0079] 因此,在第一循环i期间,至少一个数据集AA0_3、AA4_7、AA8_11和/或AA12_15下载到至少一个寄存器11。在图中的实例中,数据集AA0_3、AA4_7、AA8_11和/或AA12_15中的每一个下载到相应寄存器11,也就是说,彼此分离的四个寄存器11。寄存器11中的每一个至少暂时指派到相应ALU 9,此处分别参考为ALU 0、ALU 1、ALU2和ALU 3。在此同一个循环i期间,ALU 9可能已经实施计算操作。

[0080] 在第二循环ii期间,每一ALU 9实施计算操作,存储在相应寄存器11上的数据项中的至少一个为所述计算操作形成操作数。举例来说,ALU 0实施对于其操作数中的一个为A0的计算操作。A1、A2和A3可在第二循环ii期间未使用。

[0081] 一般来说,将数据从存储器13下载到寄存器11消耗比借助于ALU 9实施计算操作少的计算时间。因此,通常可以认为,存储器存取操作(此处读取操作)消耗单个计算循环,而借助于ALU 9实施计算操作消耗一个计算循环或一系列多个计算循环,例如四个。

[0082] 在图3的实例中,存在指派到每一ALU 9的多个寄存器11,由参考为REG A、REG B和REG C的寄存器11的群组展示。从存储器13下载到寄存器11的数据对应于群组REG A和REG B。群组REG C在此处既定存储经由由ALU 9实施的计算操作获得的数据(在写入操作期间)。

[0083] 群组REG B和REG C的寄存器11可因此含有类似于REG A的寄存器而参考的数据集：

[0084] -群组REG B包括四个寄存器11,在上面分别存储由数据B0到B3组成的数据集BB0_3、由数据B4到B7组成的数据集BB4_7、由数据B8到B11组成的数据集BB8_11,以及由数据B12到B15组成的数据集BB12_15;

[0085] -群组REG C包括四个寄存器11,在上面分别存储由数据C0到C3组成的数据集CC0_3、由数据C4到C7组成的数据集CC4_7、由数据C8到C11组成的数据集CC8_11,以及由数据C12到C15组成的数据集CC12_15。

[0086] 在图3的实例中,数据AN和BN构成用于由ALU 9实施的计算操作的操作数,而数据

项CN构成结果,其中“N”为0和15之间的整数。举例来说,在加法的情况下,CN＝AN+BN。在此实例中,由装置1实施的数据处理操作对应于16个操作。所述16个操作在如下意义上彼此独立:不需要16个操作的结果中的任一个来实施另外15个操作中的一个。

[0087] 因此,处理操作(所述16个操作)的实施可例如如下分解为18个循环。

[0088] 实例1:

[0089] -循环#0:读取AA0_3;

[0090] -循环#1:读取BB0_3;

[0091] -循环#2:计算C0(从集合CC0_3)和读取AA4_7(形成例如循环i);

[0092] -循环#3:计算C1(从集合CC0_3)和读取BB4_7(形成例如循环i);

[0093] -循环#4:计算C2(从集合CC0_3);

[0094] -循环#5:计算C3(从集合CC0_3)和写入CC0_3;

[0095] -循环#6:计算C4(从集合CC4_7)和读取AA8_11(形成例如循环ii);

[0096] -循环#7:计算C5(从集合CC4_7)和读取BB8_11(形成例如循环ii);

[0097] -循环#8:计算C6(从集合CC4_7)(形成例如循环ii);

[0098] -循环#9:计算C7(从集合CC4_7)和写入CC4_7(形成例如循环ii);

[0099] -循环#10:计算C8(从集合CC8_11)和读取AA12_15;

[0100] -循环#11:计算C9(从集合CC8_11)和读取BB12_15;

[0101] -循环#12:计算C10(从集合CC8_11);

[0102] -循环#13:计算C11(从集合CC8_11)和写入CC8_11;

[0103] -循环#14:计算C12(从集合CC12_15);

[0104] -循环#15:计算C13(从集合CC12_15);

[0105] -循环#16:计算C14(从集合CC12_15);

[0106] -循环#17:计算C15(从集合CC12_15)和写入CC12_15。

[0107] 因而应理解,除初始循环#0和#1之外,存储器存取操作(读取和写入操作)与计算操作并行地实施,而不消耗额外计算循环。读取含有(多个)数据或数据块的数据集而非读取单个数据项,使得有可能甚至在数据变为计算操作所必需的操作数之前结束从存储器13将所述数据导入到寄存器。

[0108] 在上述循环#2的实例中,如果仅立即必需的数据项(A0)将已经被读取而非读取集合AA0_3＝{A0;A1;A2;A3},则将已经必须随后实施三个额外读取操作来获得A1、A2和A3。

[0109] 为了更好地理解,且为了进行比较,下文再现其中每次读取单个数据项的处理操作的实施,而非含有(多个)数据的数据集。观察到,必需48个循环。

[0110] 实例0:

[0111] -循环#0:读取A0;

[0112] -循环#1:读取B0;

[0113] -循环#2:计算C0和写入C0;

[0114] -循环#3:读取A1;

[0115] -循环#4:读取B1;

[0116] -循环#5:计算C1写入C1;

[0117] -…

[0118] -循环#45:读取A15;

[0119] -循环#46:读取B15;

[0120] -循环#47:计算C15和写入C15。

[0121] 在实例1中(18个循环),应注意,前两个循环#0和#1构成初始化循环。初始化循环的数目I对应于每计算操作的操作数的数目。接下来,四个连续循环的型式重复四次。举例来说,循环#2到#5一起形成某一型式。每型式的循环数目对应于每数据集的数据的数目D,而型式的数目对应于待处理的数据集的数目E。因此,循环的总数目可表达如下:I+D*E。

[0122] 实现良好性能相当于将循环的总数目减少到最小值。在考虑中的条件下(也就是说,各自能够经由一个循环实施的16个基础的独立操作),因此,最佳循环数目似乎等于基础操作的数目(16)加上初始化阶段(2个循环),也就是说总共18个循环。

[0123] 在一个变型中,认为单个循环中可存取(在读取模式中或在写入模式中)的数据的数目(每数据集的数据的数目D)等于三个(而非四个),例如这是归因于硬件局限性。循环的序列接着可例如分解如下:

[0124] -2个循环的初始化阶段;以及接着

[0125] -用于待执行的16个中的总共15个基础计算操作的3个循环的5个型式;以及接着

[0126] -用于计算和记录最后基础计算操作的结果的最终循环。

[0127] 实例2:

[0128] -循环#0:读取AA0_2＝{A0;A1;A2};

[0129] -循环#1:读取BB0_2＝{B0;B1;B2};

[0130] -循环#2:计算C0(从集合CC0_2＝{C0;C1;C2})和读取AA3_5(形成例如循环i)

[0131] -循环#3:计算C1(从集合CC0_2)和读取BB3_5(形成例如循环i);

[0132] -循环#4:计算C2(从集合CC0_2)和写入CC0_2;

[0133] -循环#5:计算C3(从集合CC3_5)和读取AA6_8(形成例如循环ii);

[0134] -循环#6:计算C4(从集合CC3_5)和读取BB6_8(形成例如循环ii);

[0135] -循环#7:计算C5(从集合CC3_5)和写入CC3_5(形成例如循环ii);

[0136] -循环#8:计算C6(从集合CC6_8)和读取AA9_11;

[0137] -循环#9:计算C7(从集合CC6_8)和读取BB9_11;

[0138] -循环#10:计算C8(从集合CC6_8)和写入CC6_8;

[0139] -循环#11:计算C9(从集合CC9_11)和读取AA12_14;

[0140] -循环#12:计算C10(从集合CC9_11)和读取BB12_14;

[0141] -循环#13:计算C11(从集合CC9_11)和写入CC9_11;

[0142] -循环#14:计算C12(从集合CC12_14)和读取A15(形成例如循环i);

[0143] -循环#15:计算C13(从集合CC12_14)和读取B15(形成例如循环i);

[0144] -循环#16:计算C14(从集合CC12_14)和写入CC12_14;

[0145] -循环#17:计算C15(隔离的数据项)和写入C15(形成例如循环ii)。

[0146] 在实例2中,观察到,每一循环包含存储器存取操作(在读取模式中或在写入模式中)。因此,应理解,如果单个循环中可存取的数据的数目D严格地小于三个,则将必需额外循环来执行存储器存取操作。因此,将不再实现用于16个基础操作的最佳18个循环。然而,即使未实现最佳,循环的数目保持明显低于实例0中必需的循环数目。其中数据集包括两个

数据项的实施例展现优于当前存在的实施例的改进。

[0147] 在实例1中,如果循环#2和/或#3对应于例如如上定义的循环i,则循环#6、#7、#8和#9中的每一个对应于循环ii。当然,此可在型式之间换位。在实例2中,如果循环#2和/或#3对应于例如如上定义的循环i,则循环#5、#6和#7中的每一个对应于循环ii。当然,此可在型式之间换位。

[0148] 在至此描述的实例(特定来说,实例1和2)中,特别地实现低循环总数目,因为每含有(多个)数据的数据集实施最大数目的存储器存取操作,而非在单元处且与计算操作并行实施。因此,对于过程的一些部分(对于经优化实例中的所有部分),可甚至在先前基础计算操作已经结束之前实现对所有必需的操作数的读取操作。优选地节省了计算能力以便在共同计算循环(例如,在实例1中,循环#5)中执行计算操作和记录(写入操作)所述计算操作的结果。

[0149] 在所述实例中,在整个过程中实施操作数数据的先进读取(从一个型式到另一型式重复)。在时间上前一型式期间自动获得(读取)某一型式期间执行的计算操作所必需的操作数。应注意,在降级的实施例中,仅部分(仅针对两个连续型式)实施先进读取。相比于以上实例降级的此模式比现有方法展现更好的结果。

[0150] 在至此描述的实例中,已认识到,数据在充当操作数之前被读取。在一些实施例中,事先读取的数据被随机读取,或至少独立于待执行的将来计算操作而读取。因此,数据集当中的至少一些事先读取的数据有效地对应于用于后续计算操作的操作数,而其它读取数据不是用于后续计算操作的操作数。举例来说,至少一些所读取数据可在未被ALU9使用的情况下随后从寄存器11擦除,通常被随后记录在寄存器11上的其它数据擦除。因此,不必要地读取了一些数据(且其被不必要地记录在寄存器11上)。然而,来自被读取数据集的至少一些数据有效地为操作数以便实现计算循环的节省已经足够,且因此情形相比于当前存在的情形得以改进。因此,取决于待处理的数据的数目和循环的数目,很可能(在所述术语的数学意义上)至少一些预取的数据将有效地能够在随后循环中由ALU 9执行的计算操作中用作操作数。

[0151] 在一些实施例中,取决于待执行的计算操作预先选择事先读取的数据。这使得有可能改进预取的数据的相关性。确切地说,在上述具有16个基础计算操作的实例中,16个基础计算操作中的每一个在输入处需要一对操作数,分别为A0和B0;A1和B1;……;A15和B15。如果随机读取数据,则两个第一循环可对应于AA0_3和BB4_7上的读取操作。在此情况下,在前两个循环结束时寄存器11上没有完整的操作数对可用。因此,ALU 9不能够在随后循环中实施任何基础计算操作。因此,在基础计算操作能够开始之前,将必定消耗一个或多个额外循环用于存储器存取操作,借此增加循环的总数目且对效率产生不利影响。

[0152] 对读取模式中获得的数据尽可能相关的机率和概率进行计数足以改进当前存在的情形,但不完全令人满意。所述情形能够进一步改进。

[0153] 实施预取算法使得有可能尽可能早地获得待执行的下一计算操作的所有操作数。在以上实例中,在前两个循环期间读取AA0_3和BB0_3使得有可能例如使实施4个第一基础计算操作必需的所有操作数在寄存器11上可用。

[0154] 此算法接收关于随后待由ALU 9执行的计算操作且确切地说关于必需的操作数的信息数据,作为输入参数。此算法使得有可能预期待执行的将来计算操作而在输出处选择

所读取(每集合)的数据。此算法例如由控制单元5在控制存储器存取操作时实施。

[0155] 根据第一方法,一旦数据记录在存储器13中,算法就强加数据的组织。举例来说,需要形成数据集的数据经并置和/或排序使得整个数据集能够通过单个请求调用。举例来说,如果数据A0、A1、A2和A3的地址分别参考为@A0、@A1、@A2和@A3,则可响应于@A0上的读取请求来配置存储器接口15,以便还自动读取随后三个地址@A1、@A2和@A3处的数据。

[0156] 根据第二方法,预取算法在输出处提供存储器存取请求,所述存储器存取请求基于随后待由ALU 9执行的且确切地说与必需的操作数相关的计算操作来调适。在上述实例中,举例来说,算法识别出优先读取的数据是AA0_3和BB0_3的数据,以便较早地在随后循环中启用基础计算操作,给出结果CC0_3,也就是说,以操作数A0和B0计算C0,以操作数A1和B1计算C1,以操作数A2和B2计算C2,以及以操作数A3和B3计算C3。因此,算法在输出处提供被构造以便生成AA0_3和BB0_3上的读取操作的存储器存取请求。

[0157] 两种方法可任选地彼此组合:算法识别待读取的数据且控制单元5自其推断存储器接口15处的存储器存取请求以便获得所述数据,所述请求基于存储器接口15的特征(结构和协议)来调适。

[0158] 在上述实例(特定来说,上文的实例1和2)中,不限定指派到基础计算操作的ALU的数目。单个ALU 9可逐个循环地执行所有基础计算操作。待执行的基础计算操作还能够分布在PU的多个ALU 9上,例如四个。在此些情况下,利用将待在每一读取操作中读取的数据分组在一起的技术协调计算操作在ALU上的分布可使得有可能进一步改进效率。在两种方法之间作出区分。

[0159] 在第一方法中,在操作中读取的数据形成由仅同一个ALU 9实施的计算操作中的操作数。举例来说,首先读取数据A0、A1、A2、A3、B0、B1、B2和B3的群组AA0_3和BB0_3,且使第一ALU负责计算CC0_3(C0、C1、C2和C3)。接着读取群组AA4_7(A4、A5、A6、A7)和BB4_7(B4、B5、B6和B7),且使第二ALU负责计算CC4_7(C4、C5、C6和C7)。因而应理解,第一ALU将能够在第二ALU能够实施计算操作之前开始实施计算操作,因为第一ALU的计算操作所必需的操作数将在第二ALU的计算操作所必需的操作数之前在寄存器11上可用。PU的ALU 9接着并行地且异步地操作。

[0160] 在第二方法中,在操作中读取的数据形成各自由不同ALU 9(例如,四个)实施的计算操作中的操作数。举例来说,首先读取分别包含A0、A4、A8和A12;B0、B4、B8和B12的数据的两个群组。使第一ALU负责计算C0,使第二ALU负责计算C4,使第三ALU负责计算C8,且使第四ALU负责计算C12。因而应理解,四个ALU将能够大体上同时开始实施其相应计算操作,因为必需的操作数将与其在共同操作中下载同时在寄存器11上可用。PU的ALU 9并行且同步地操作。取决于待执行的计算操作的类型、存储器中的数据的可及性和可用资源,两种方法中的一种或另一种可以是优选的。两种方法还可组合:ALU可组织成子群组,子群组的ALU同步操作,且子群组相对于彼此异步地操作。

[0161] 为了强加ALU的同步、异步或混合操作,每读取操作待读取的数据在一起的分组应选择成对应于计算操作向各个ALU的指派的分布。

[0162] 在上述实例中,基础计算操作彼此独立。因此,据推断,其执行次序不具有任何重要性。在对于其至少一些计算操作彼此相依的一些应用中,计算操作的次序可为特定的。此情形通常在递归计算操作的上下文中发生。在此些情况下,算法可配置成将待获取(读取)

的数据识别为优先级。举例来说,如果:

[0163] -经由其一个操作数为C0的计算操作获得结果C1,C0本身是从操作数A0和B0获得,

[0164] -经由其一个操作数为C4的计算操作获得结果C5,C4本身是从操作数A4和B4获得,

[0165] -经由其一个操作数为C8的计算操作获得结果C9,C8本身是从操作数A8和B8获得,以及

[0166] -经由其一个操作数为C12的计算操作获得结果C13,C12本身是从操作数A12和B12获得,

[0167] 则算法可配置成在前两个初始化循环#0和#1期间读取如下限定的数据集:

[0168] -{A0;A4;A8;A12},以及

[0169] -{B0;B4;B8;B12}。

[0170] 如此限定的数据集在图4中展示。形象地,可陈述,数据在图3中展示的实施例中“成行”分组在一起,且在图4中展示的实施例中“成列”分组在一起。如此实施算法使得有可能读取可用于优先级基础计算操作的操作数,且使其在寄存器11上可用。换句话说,实施所述算法使得有可能相比于随机读取操作增加所读取数据的短期相关性。

[0171] 本发明不限于上文仅借助于实例描述的处理单元和方法的实例,而是并入有所属领域的技术人员将能够在所寻求的保护范围内预期的所有变型。本发明还涉及用于获得此计算装置(例如,处理器或处理器的集合)的处理器可实施机器指令的集合,涉及此机器指令集合在处理器上的实施,涉及由处理器实施的处理器架构管理方法,涉及包括所述相应机器指令集合的计算机程序,且涉及上面以计算方式记录此机器指令的集合的记录介质。

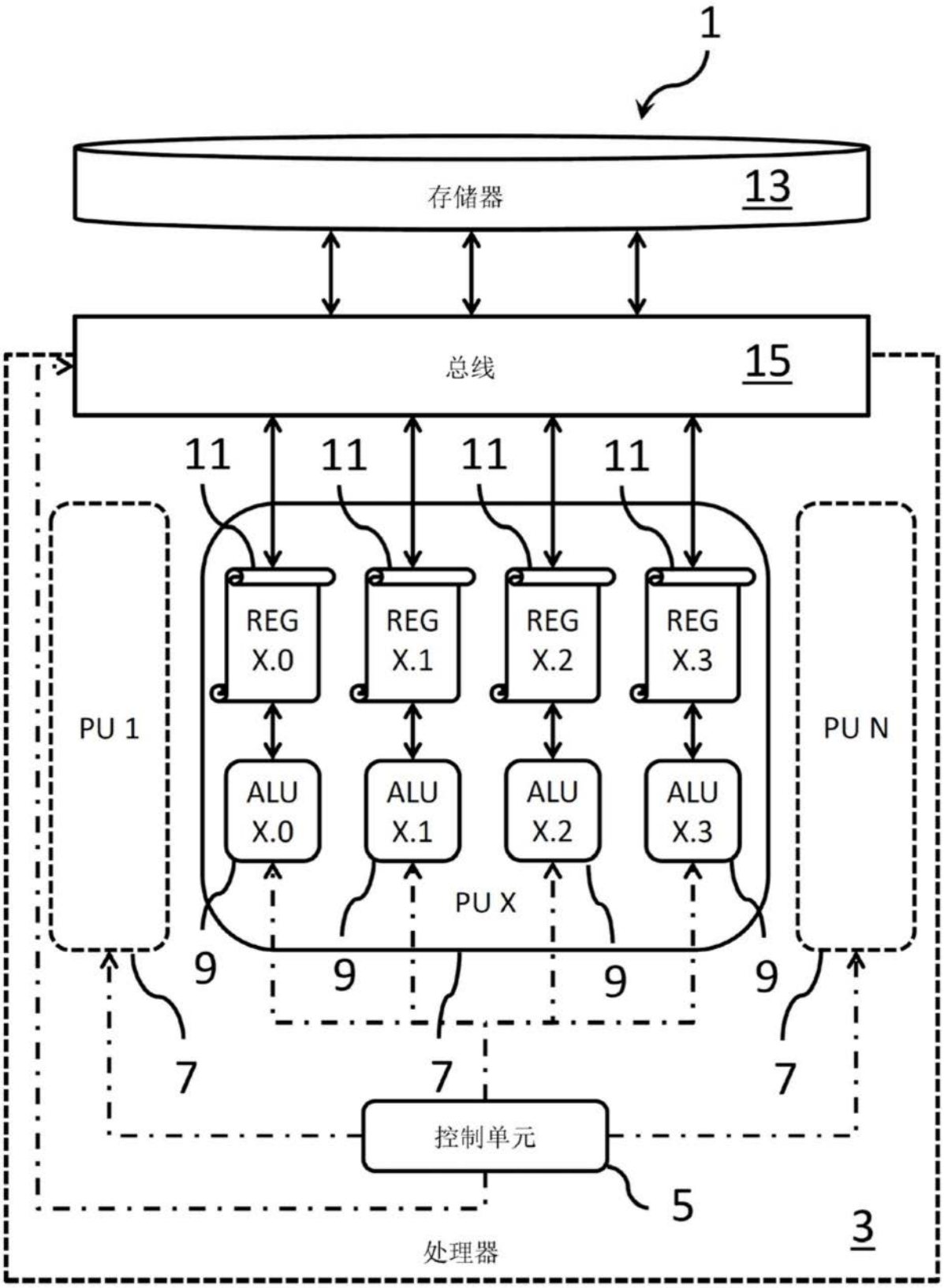

1
存储器
13
总线
15
11
11
11
11
REG X.0
REG X.1
REG X.2
REG X.3
ALU X.0
ALU X.1
ALU X.2
ALU X.3
PU 1
PU X
PU N
9
9
9
9
7
7
7
控制单元
5
3
处理器

图1

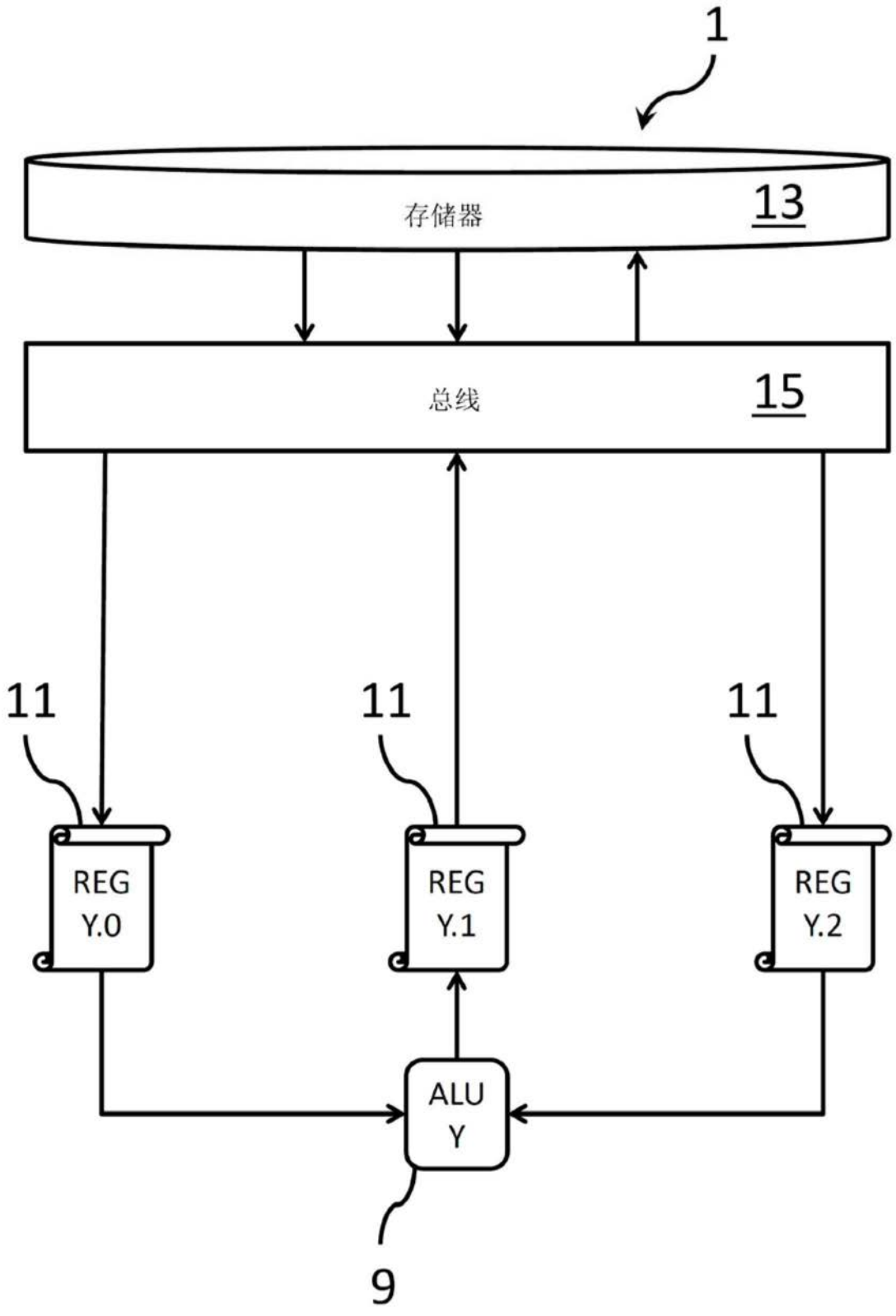


图2

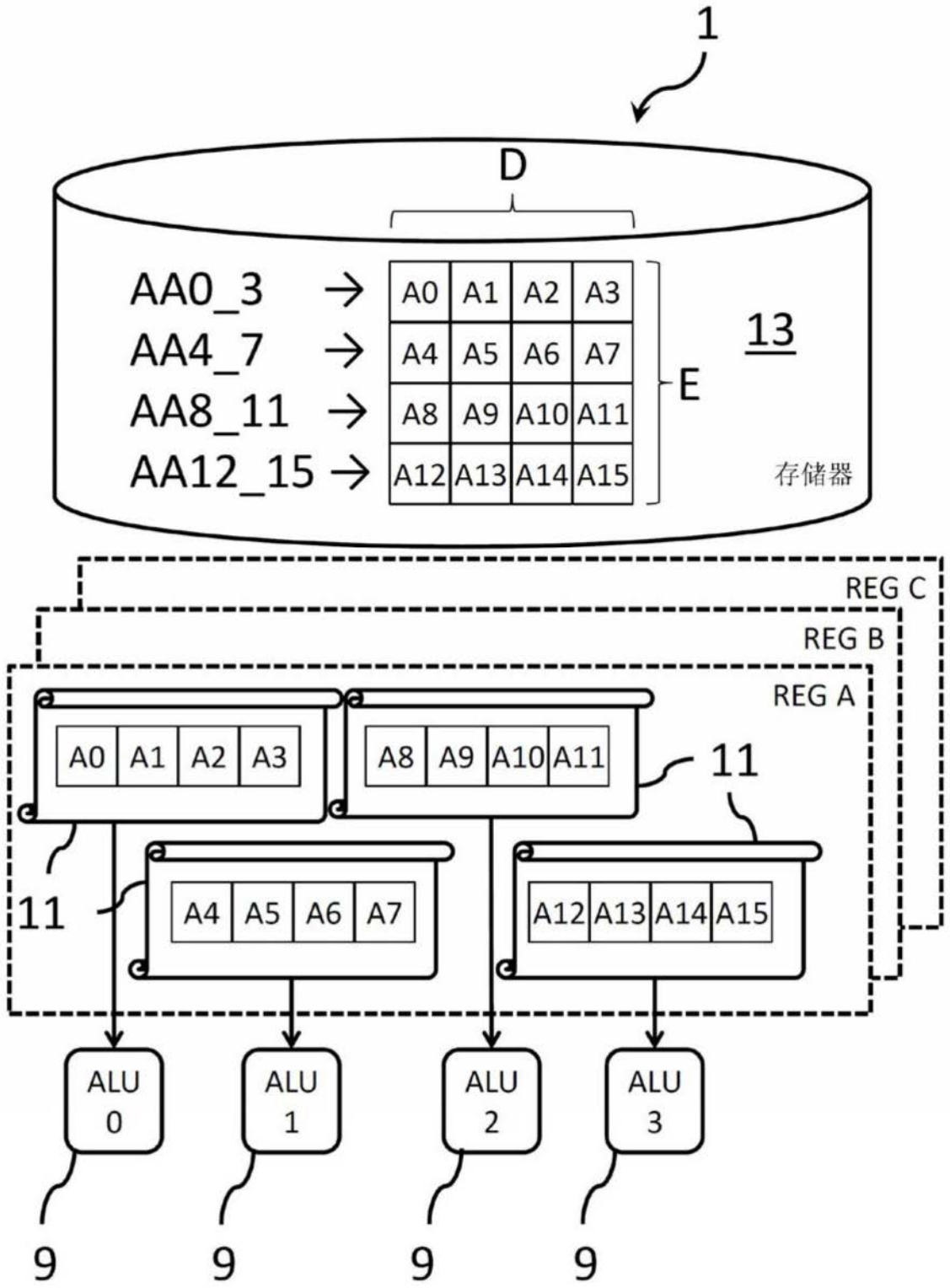

1
D
AA0_3
AA4_7
AA8_11
AA12_15
A0 A1 A2 A3
A4 A5 A6 A7
A8 A9 A10 A11
A12 A13 A14 A15
E
13
存储器
REG C
REG B
REG A
11
11
ALU 0
ALU 1
ALU 2
ALU 3
9
9
9
9

图3

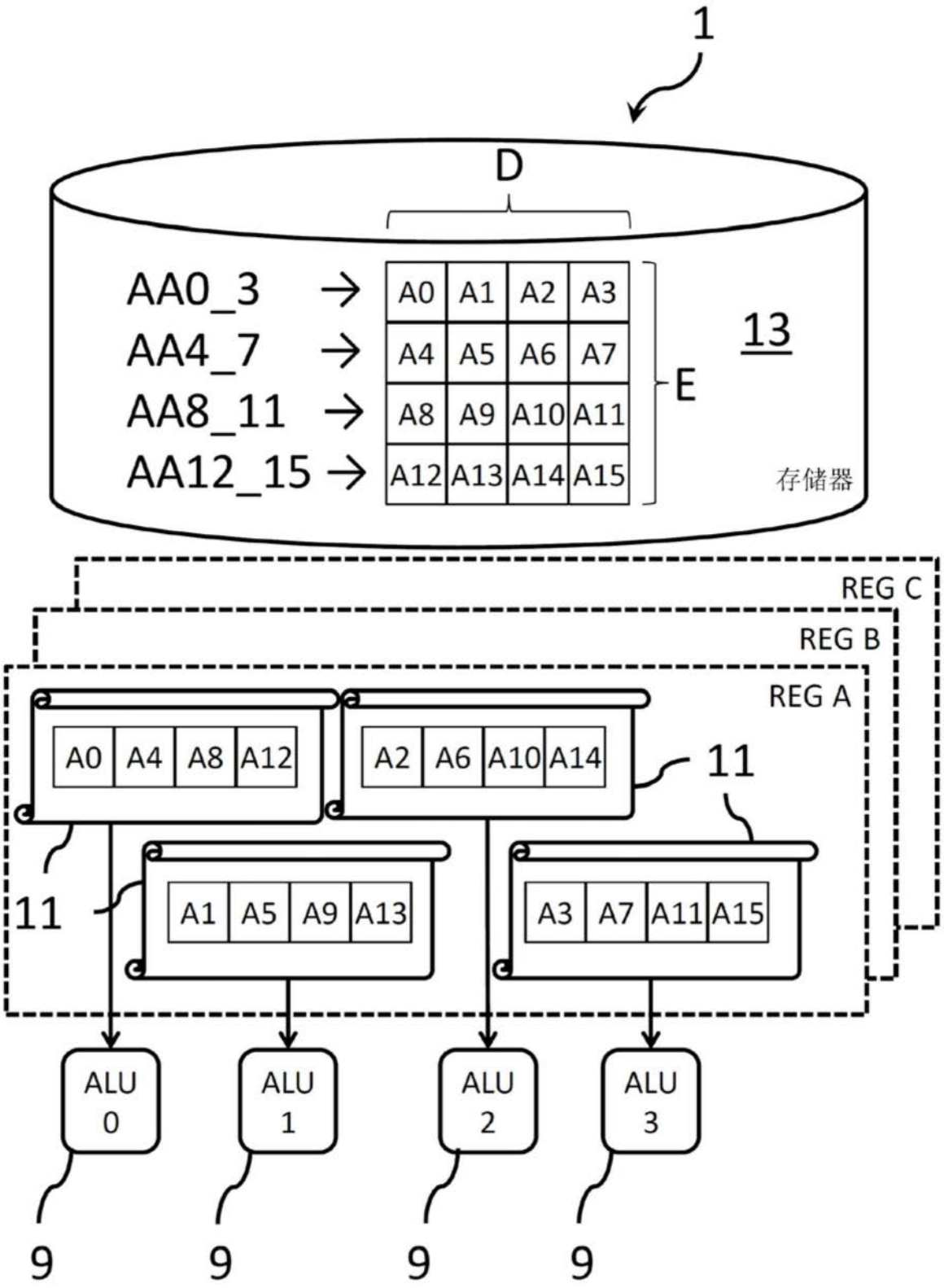

1
D
AA0_3
AA4_7
AA8_11
AA12_15
A0 A1 A2 A3
A4 A5 A6 A7
A8 A9 A10 A11
A12 A13 A14 A15
E
13
存储器
REG C
REG B
REG A
A0 A4 A8 A12
A2 A6 A10 A14
11
11
A1 A5 A9 A13
A3 A7 A11 A15
ALU 0
ALU 1
ALU 2
ALU 3
9
9
9
9

图4